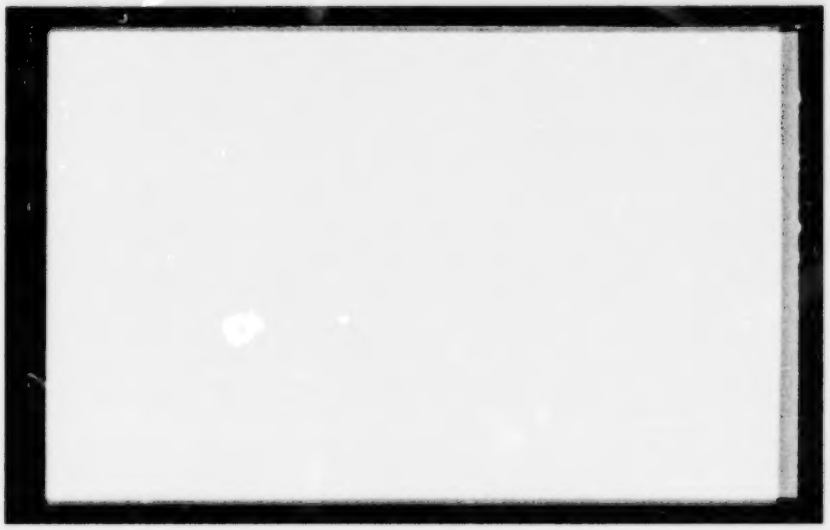
UNIVERSITÉ DE MONTRÉAL



CENTRE DE RECHERCHE SUR LES TRANSPORTS

CASE POSTALE 6128, STATION A, MONTRÉAL, QUÉBEC, H3C 3J7

MODIFICATION ET IMPLANTATION D'UN MODELE LINEAIRE POUR UNE PLANIFICATION A MOYEN TERME D'UNE FLOTTE D'AVIONS

PAR

André Daigle

Université de Montréal
Publication # 241 - Département d'informatique
Publication # 48 - Centre de recherche sur les transports
Mars 1976

TABLE DES MATIERES

Chapitre		Page
	TABLE DES MATIERES	iii
	TABLE DES FIGURES	iv
	SOMMAIRE	v
I	INTRODUCTION	1
II	LE MODELE ORIGINAL	3
	2.1 Présentation du problème	3
	2.2 Etude du modèle	7
III	MODIFICATIONS DU MODELE	16
	3.1 Définition des ensembles	18
	3.2 Modifications du modèle lui-même	18
	3.3 Commentaires généraux	24
IV	TECHNIQUES D'IMPLANTATION ET DOCUMENTATION	28
	4.1 Représentation des routes et des routes à connexions	28
	4.2 Génération des ensembles	32
	4.3 Génération des contraintes	40
	4.4 Programme principal	45
	REMERCIEMENTS	47
	BIBLIOGRAPHIE	48
	ANNEXE: LE PROGRAMME ET SES ROUTINES	49

TABLE DES FIGURES

		Page
Figure 1	Les routes de VOLEUSE	4
Figure 2	Courbe de fréquence vs demande	4
Figure 3	Représentation des coefficients de remplissage sur la courbe de fréquence vs demande	5
Figure 4	Ensemble de 4 routes desservant le marché p-q	7
Figure 5	Courbe segmentée de fréquence vs demande	10
Figure 6	Route à connexions unique	17
Figure 7	Courbe segmentée de fréquence vs demande dans le modèle modifié	20
Figure 8	Représentation des vecteurs VILLE, IPT, X et IX	29
Figure 9		29
Figure 10		30
Figure 11		30
Figure 12		31
Figure 13	Relations entre le programme principal et ses routines	46

SOMMAIRE

Ce rapport de stage analyse un modèle mis au point au "FLIGHT TRANSPORTATION LABORATORIES", de M.I.T.; ce modèle linéaire vise à obtenir une bonne planification à moyen et à long terme d'une flotte d'avions dans un réseau aérien.

Ensuite, ce modèle fut modifié pour tenir compte des connexions, une réalité importante dans la plupart des compagnies d'aviation dont AIR CANADA.

Finalement, ce rapport de stage détaille les algorithmes de l'implantation du modèle modifié sur ordinateur. Cette implantation fut réalisée en partant des données d'AIR CANADA.

Ce présent rapport ne précise aucun résultat puisque des études de calibration du modèle de même que des études de sensibilité doivent être exécutées à la suite de ce stage.

CHAPITRE I

INTRODUCTION

Le stage avait pour objectif d'analyser le modèle qui avait été conçu au "FLIGHT TRANSPORTATION LABORATORIES " de M.I.T. tel que décrit dans "THE COMPLEAT FA-4 MEMO" de William Swan [1].

Ce modèle vise à obtenir une bonne planification à moyen et à long terme d'une flotte d'avions dans un réseau aérien; ce modèle maximise la différence entre les revenus et les coûts d'opération dans le cadre d'un modèle linéaire.

Le modèle fut modifié afin de tenir compte des connexions, une réalité importante dans la plupart des compagnies d'aviation dont AIR CANADA.

Enfin, il fallait implanter le modèle modifié sur ordinateur et ce, avec les données d'AIR CANADA. A la suite de ce stage, des études de calibration du modèle, des études de sensitivité de même qu'une analyse des résultats devront être effectuées.

Ce rapport se divise en 3 parties. La deuxième section présente le modèle original; dans la troisième section, on retrouve les modifications théoriques qui ont été élaborées et enfin la quatrième section explicite les différentes techniques d'implantation.

Il est bon de mentionner tout de suite que le résultat final consiste à créer un fichier d'entrée qui est traité par le programme produit APEX-II[2], lequel a été conçu pour résoudre des problèmes linéaires de grande

échelle. APEX-II utilise la méthode du simplexe révisé avec la forme du produit de l'inverse. Des modifications mineures permettent l'utilisation de ce fichier d'entrée avec des programmes produits similaires à APEX-II.

CHAPITRE II

LE MODELE ORIGINAL

Nous faisons ressortir dans cette section les idées de base qui soutiennent le modèle original. Dans cet optique, on présente un exemple simple qui met en évidence la nature du problème. Ensuite, on présente le modèle lui-même avec chacune de ses contraintes et la fonction objective.

Il est nécessaire de comprendre ce qui se passe au niveau de ce modèle afin de saisir la portée des modifications proposées dans la troisième section.

2.1 Présentation du problème

Une route est un parcours d'avion. Il peut y avoir des routes sans arrêt, à 1 arrêt ou à 2 arrêts. Une route sera dénotée par la suite des villes qu'elle dessert.

Supposons qu'une compagnie d'aviation appelée "VOLEUSE" desserve 4 villes A, B, C, D. Les routes que VOLEUSE dessert sont illustrées à la figure 1.

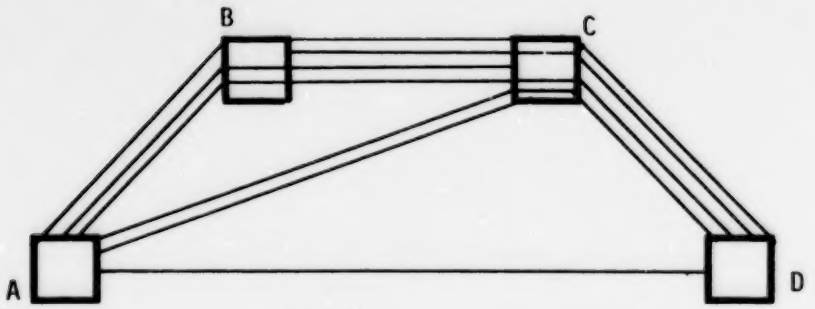


FIGURE 1 LES ROUTES DE VOLEUSE

L'ensemble des routes, au nombre de 9, est dénoté par:

R = {AD, AB, AC, ABC, ABCD, ACD, BC, BCD, CD}

Supposons aussi qu'il y ait un seul type d'avions:

AV40 —— 40 passagers

VOLEUSE fonctionne sur le principe que pour chaque couple origine-destination, le nombre de passagers qu'elle transporte est dépendant de la fréquence de service offert à ce couple origine-destination. Cette dépendance est illustrée sur la figure 2.

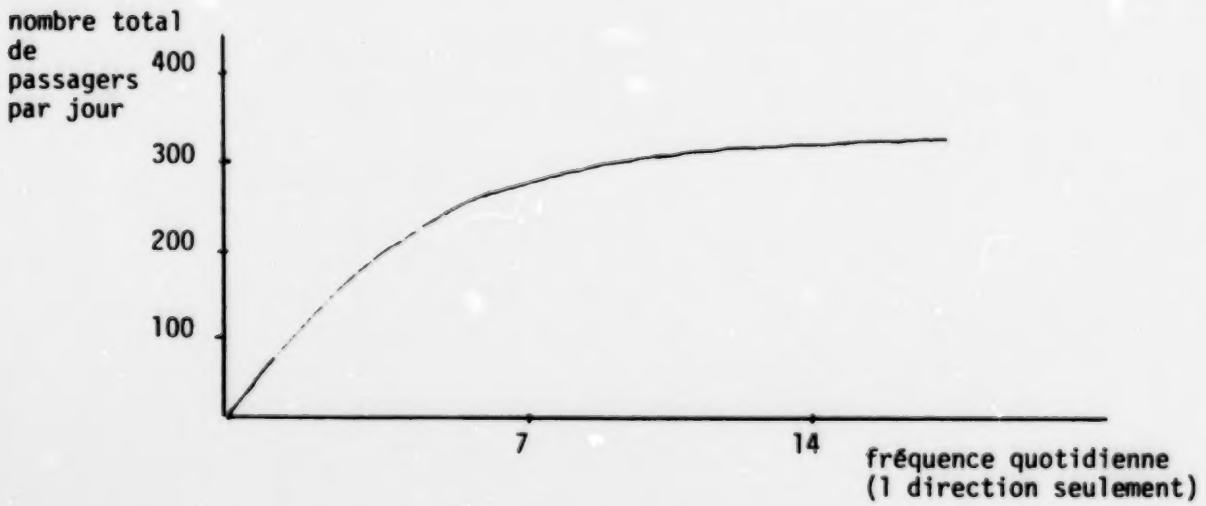


FIGURE 2 COURBE DE FREQUENCE VS DEMANDE

Connaissant les demandes de marché pour chaque couple origine-destination et les disponibilités des avions, VOLEUSE veut déterminer les routes à emprunter et le type d'avions à utiliser de façon a maximiser le profit.

VOLEUSE connait aussi ses coûts d'opération (pour chaque type d'avions sur chaque route). Ces coûts se répartissent en coûts directs d'opération dûs aux déplacements de l'avion, aux distances parcourues, aux frais des pilotes et des hôtesses ainsi qu'au coût d'atterrissage à chaque aéroport; et en coûts indirects associés au coût d'achat, de financement des avions ainsi qu'au coût fixe relié à l'administration de la compagnie.

D'autre part, VOLEUSE connait le prix des billets pour chaque couple origine-destination. Supposons que la figure 2 représente le marché A-D. La figure 3 reprend la figure 2 en ajoutant des droites. Ces droites déterminent le nombre de sièges disponibles sur AV40 en fonction de la fréquence du service offert lorsque AV40 opère avec respectivement un coefficient de remplissage maximum (avion plein) et avec un coefficient minimum pour assurer la rentabilité.

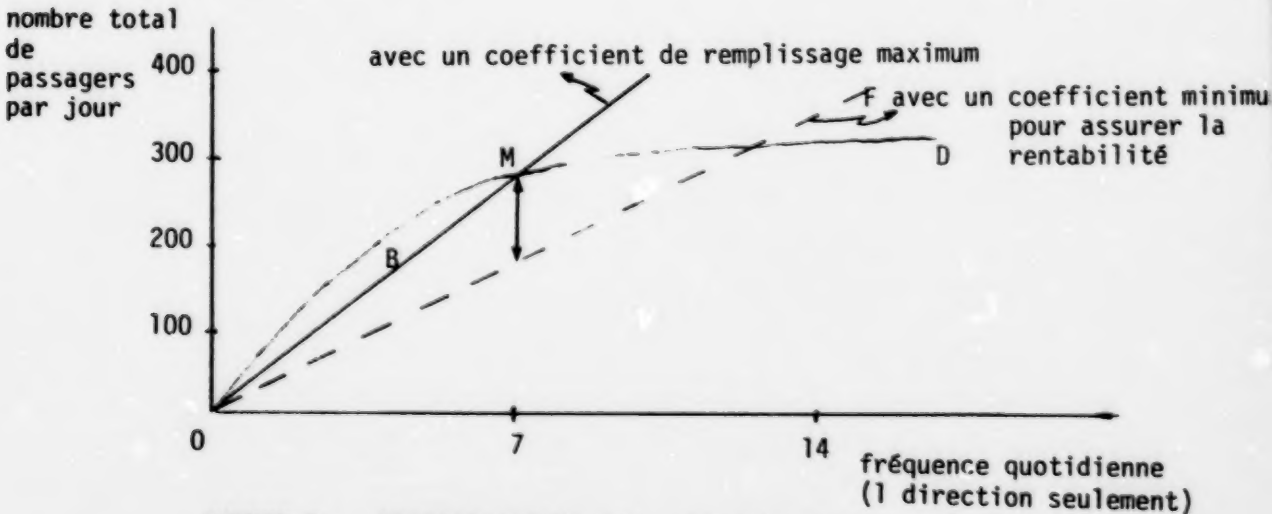


FIGURE 3 REPRESENTATION DES COEFFICIENTS DE REMPLISSAGE SUR LA COURBE DE FREQUENCE VS DEMANDE

Se référant à la figure 3, si la fréquence est inférieure à 7, alors la demande est plus grande que la disponibilité en sièges et par conséquent le nombre de passagers transportés est déterminé en utilisant la droite OBM. D'autre part, si la fréquence est supérieure à 7, alors le nombre de sièges disponibles excède la demande et par conséquent le nombre de passagers transportés est déterminé en utilisant la courbe MD.

Ainsi la courbe OBMD détermine le nombre de passagers transportés en fonction de la fréquence. Ceci correspond aux revenus. Alors, si la droite pointillée OF représente le nombre de sièges occupés lorsque AV40 opère avec un coefficient minimum pour assurer la rentabilité, il devient évident que la plus grande marge de profit au sens de revenu net maximum est obtenue avec une fréquence de 7 (sous l'hypothèse que AV40 est le seul type d'avions disponible et en l'absence de toute contrainte).

Le nombre de passagers qui désire voyager d'une origine à une destination est également influencé par le genre de service offert en plus d'être influencé par la fréquence du service. Ainsi un service direct est plus attrayant qu'un service comportant un ou plusieurs arrêts. Le type d'avions assurant le service peut également influencer le nombre de passagers désirant voyager. Ces considérations sont inclues dans la notion de "fréquence quotidienne équivalente" qui remplace la fréquence quotidienne de la figure 3. Pour calculer la fréquence quotidienne équivalente, on affecte un coefficient plus petit ou égal à 1 à chaque service, coefficient qui diminue avec le nombre d'arrêts et qui varie suivant le type d'avions. La fréquence équivalente est égale à la somme de ces coefficients.

Finalement le modèle tient compte du fait que certains types d'avions ne peuvent desservir certains couples origine-destination ou ne peuvent se poser à certains aéroports. Les aéroports ont également des contraintes de capacité qu'il faut respecter. Une borne inférieure sur la fréquence de service peut être spécifiée pour certains couples origine-destination.

2.2 Etude du modèle

Pour tout couple origine-destination, on doit disposer de la courbe du nombre de passagers en fonction de la fréquence équivalente. Cette courbe sera brisée en 3 segments de droite. (Voir figure 5).

Formellement, une route est un itinéraire reliant deux villes p et q pouvant comporter un certain nombre de villes intermédiaires. Chaque ville intermédiaire correspond à un arrêt qu'on doit faire si on utilise la route pour se rendre de p à q .

Dans la figure 4:

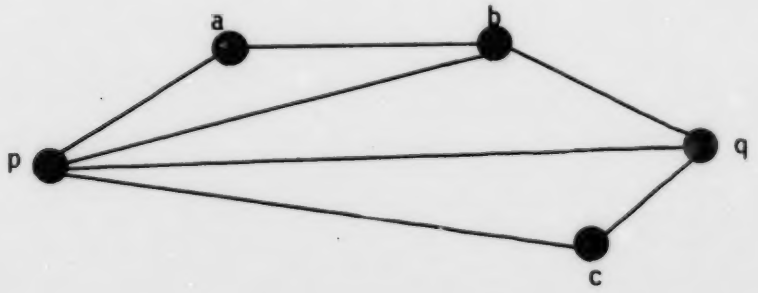


FIGURE 4 ENSEMBLE DE 4 ROUTES DESSERVANT LE MARCHE p-q

On retrouve les routes p-a-b-q, p-c-q, p-b-q, p-q

Un segment est une partie de route ne comportant aucune ville intermédiaire (par exemple, dans la figure 4: p-a, a-b, c-q sont des segments).

On définit les ensembles suivants:

- R = {toutes les routes du réseau}
- R_{pqm} = {toutes les routes reliant p à q en exactement m arrêts} $(R_{pqm} \subset R)$
- $R_{pq} = \bigcup_{m} R_{pqm}$ $(R_{pq} \subset R)$
- R_{ij} = {toutes les routes reliant i à j} $(R_{ij} \subset R)$

Notons que tout segment i-j constitue un marché p-q mais que la réciproque n'est pas vraie.

2.2.1 Liste des variables

pq : le couple origine-destination définissant le marché

ij : segment entre les villes i et j (sans arrêt)

r : une route

m : l'indice désignant le nombre d'arrêts sur une route

a : l'indice désignant le type d'avions

K : une division de la courbe de fréquence segmentée (voir figure 5)

Y_{pq} : le prix d'un billet pour un passager sur le marché pq

P_{pqr} : le nombre de passagers du marché pq utilisant la route r

DC_{ra} : le coût direct d'opération pour un avion de type a sur la route r

N_{ra} : le nombre de services exécutés par un avion de type a sur la route r

LF_{ra} : le coefficient de remplissage maximum pour un avion de type a sur la route r

S_a : le nombre de sièges d'un avion de type a

K_m : le coefficient de fréquence pour un service comportant m arrêts

p^K_{pq} : la pente dans la région de fréquence K de la courbe segmentée du marché pq (voir figure 5)

n^K_{pq} : la fréquence quotidienne équivalente pour un segment K de la courbe segmentée du marché pq (voir figure 5) .

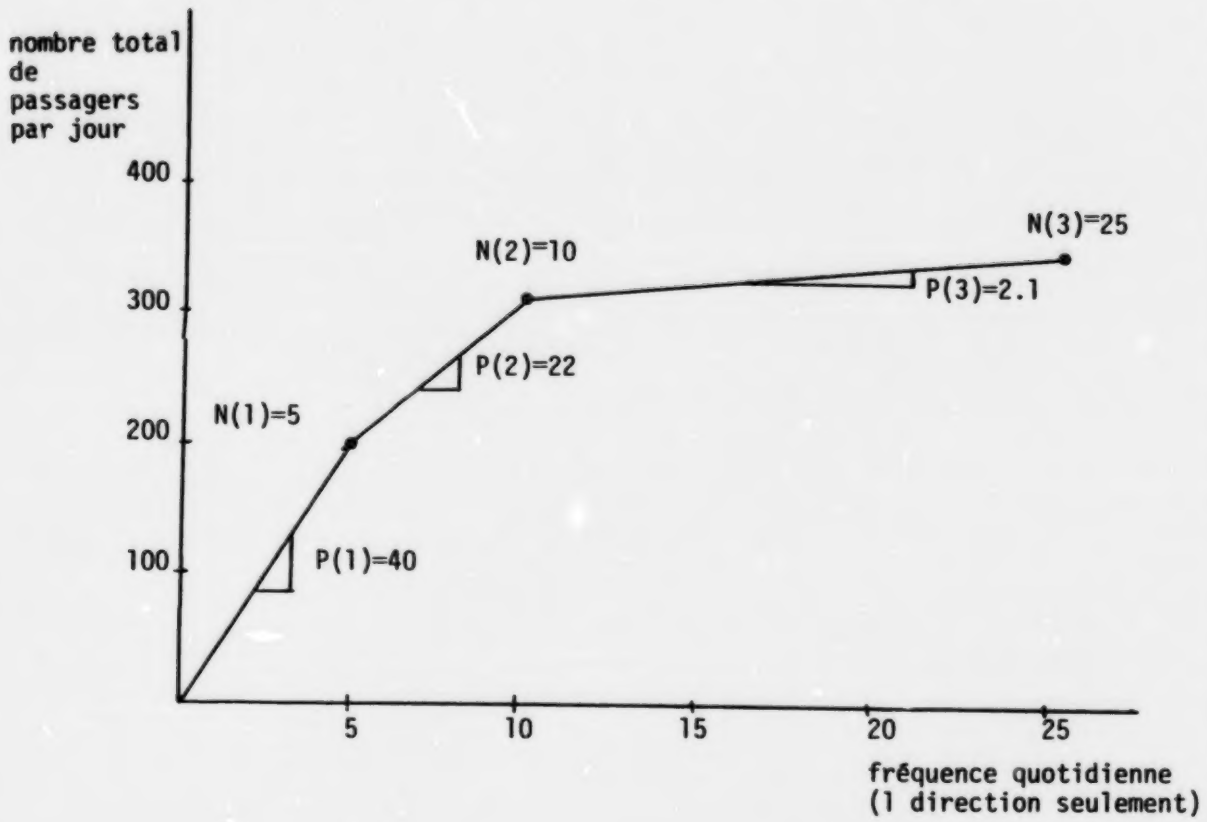


FIGURE 5 COURBE SEGMENTEE DE FREQUENCE VS DEMANDE

La figure 5 représente une courbe de fréquence segmentée en 3 tronçons (K = 3) pour un marché p-q fixé.

Donc: $P(K) = P_{pq}^{K}$

$N(K) = N_{pq}^{K}$

2.2.2 Le modèle mathématique

$$\underline{\text{MAX}} \left\{ \sum_{pq} \sum_{r \in R_{pq}} Y_{pq} \cdot P_{pqr} - \sum_{r} \sum_{a} DC_{ra} \cdot N_{ra} - IC_{t} \right\}$$

S.A.

1) $$\sum_{a} LF_{ra} \cdot S_{a} \cdot N_{ra} - \sum_{\substack{p<j \\ i<q}} P_{pqr} \geq 0 \quad , \forall ij, \forall r$$

2) $$\sum_{K} P^{K}_{pq} \cdot n^{K}_{pq} - \sum_{r \in R_{pq}} P_{pqr} \geq 0 \quad , \forall pq$$

3) $$\sum_{m=0,1,2} \sum_{a} \sum_{r \in R^{K_m}_{pqm}} N_{ra} - \sum_{K} n^{K}_{pq} = 0 \quad , \forall pq$$

4) $$\sum_{r} Tb_{ra} \cdot N_{ra} \leq U_{a} \cdot A'_{a} \quad , \forall a$$

5) $$\sum_{m=0,1,2} \sum_{r \in R_{pqm}} \sum_{a} N_{ra} \geq NMmin_{pq} \quad , \forall pq$$

6) $$\sum_{a} \sum_{j} \sum_{r \in R_{ij}} N_{ra} \leq Nmax_{i} \quad , \forall i$$

Note: $n^{K}_{pq} \leq N^{K}_{pq}$

2.2.3 Interprétation des équations du modèle

. La fonction économique:

$$\text{MAX}\{\underbrace{\sum_{pq}\sum_{r\in R_{pq}} Y_{pq}\cdot P_{pqr}}_{\text{revenus}} - \underbrace{\sum_{r}\sum_{a} DC_{ra}\cdot N_{ra}}_{\substack{\text{coûts directs}\\ \text{d'opération}}} - \underbrace{IC_t}_{\substack{\text{variable de coûts}\\ \text{indirects pour le système}}}\}$$

. 1^{er} type de contraintes:

$$\sum_{a} LF_{ra}\cdot S_a\cdot N_{ra} - \sum_{\substack{p<j\\ i<q}} P_{pqr} \geq 0 \quad \forall ij, \forall r$$

Ce type de contraintes impose que le nombre de sièges moins le nombre de passagers soit non négatif pour tout segment ij de la route r.

. 2^{o} type de contraintes:

$$\sum_{m} K_m \sum_{a} \sum_{r\in R_{pqm}} N_{ra} - \sum_{K} n_{pq}^{K} = 0 \quad \forall pq$$

(I)

Ce type de contraintes génèrent les valeurs que prennent les variables n_{pq}^{K}.

Rappelons que les bornes N_{pq}^{K} délimitent les variables n_{pq}^{K} de la façon suivante:

$$\sum_{i=1}^{k} n_{pq}^{i} \leq N_{pq}^{k} \quad k = 1,2,...$$

Ainsi, par exemple, si $N_{pq}^{1} = 5$, $N_{pq}^{2} = 10$, $N_{pq}^{3} = 25$

et si $\sum_{m} K_m \sum_{a} \sum_{r \epsilon R_{pqm}} N_{ra} = 8$,

alors $n_{pq}^{1} = 5$, $n_{pq}^{2} = 3$ et $n_{pq}^{3} = 0$

(I) représente le nombre total de fois qu'un avion dessert le marché pq ; ce nombre est pondéré par le nombre d'arrêt(s).

$$0 < K_m \leq 1 \quad \text{avec} \quad K_{m+1} < K_m$$

c'est-à-dire: plus il y a d'arrêts, plus le poids est faible.

. 3^{o} type de contraintes:

$$\sum_{K} p_{pq}^{K} \cdot n_{pq}^{K} - \sum_{r \epsilon R_{pq}} P_{pqr} \geq 0 \quad \forall pq$$

Ce type de contraintes illustre l'hypothèse que le trafic du marché pq est fonction de la fréquence quotidienne équivalente.

. 4^{o} type de contraintes:

$$\sum_{r} Tb_{ra} \cdot N_{ra} \leq U_{a} \cdot A'_{a} \quad , \forall a$$

Cette contrainte limite le temps d'utilisation d'un avion par jour. Elle peut se lire comme suit: pour un avion de type a, la somme des produits (temps de service sur une route r par le nombre de services sur cette même route) doit être plus petit ou égal à un nombre d'heures d'utilisation maximum pour ce type d'avions dans une journée par le nombre d'avions disponibles de ce type.

Notons immédiatement que le modèle modifié opérera sur une base hebdomadaire et conséquemment les U_{a} représenteront un nombre d'heures d'utilisation maximum par semaine.

. 5^{o} type de contraintes:

$$\sum_{m=1,M} \sum_{r \in R_{pqm}} \sum_{a} N_{ra} \geq NMin_{pq} \quad , \forall pq$$

Pour tout couple origine-destination, on impose une borne inférieure sur le nombre de services.

. 6^{o} type de contraintes:

$$\sum_{a} \sum_{j} \sum_{r \in R_{ij}} N_{ra} \leq Nmax_{i} \quad , \forall i$$

Pour toute ville, on impose une borne supérieure sur le nombre de départs.

La taille du problème:

En adoptant les conventions suivantes:

$|pq|$: le nombre de couples origine-destination

$|a|$: le nombre d'avions

$|Q|$: le nombre de villes

$|R|$: le nombre de routes

$\sum_{ij} |R_{ij}|$: le nombre de segments

$\sum_{pq} |R_{pq}|$: le nombre de triplets origine-destination-route

la taille du problème peut s'exprimer de la façon suivante:

- Le nombre de contraintes $\leq 3\,|pq| + |a| + |Q| + \sum_{ij} |R_{ij}|$
- Le nombre de variables $\leq |R| \cdot |a| + \sum_{pq} |R_{pq}| + |pq| \cdot |\bar{K}|$

Ainsi, pour un problème comportant 625 villes, 200 couples origine-destination, 5 types d'avions, 500 routes, 700 segments de route, 600 triplets origine-destination-routes, $\bar{K} = 3$ segments sur la courbe de fréquence (en moyenne);

alors le nombre de contraintes est: N.C. $\leq 3(200) + 5 + 25 + 700 = 1330$

et le nombre de variables est: N.V. $\leq 5(500) + 600 + 3(200) = 3700$.

CHAPITRE III

MODIFICATIONS DU MODELE

Essentiellement, le modèle original a été modifié dans le but de tenir compte d'une réalité importante dans la plupart des compagnies d'aviation qui est celle des connexions.

Une connexion implique une escale et un changement d'avion. Ceci permet donc à un passager de se rendre de A à B avec une ou plusieurs connexions sans emprunter une route telle que définie à la section précédente. Par conséquent, certains passagers voyageront sur des routes "fictives" qui sont constituées par la concaténation de segments (parcours d'avion entre 2 villes sans arrêt). Dans un contexte de choix de routes, le coût du billet pour une telle route à connexions n'est pas différent de celui pour une route alternative.

Il devient clair que les routes à connexions ne prendront un sens que dans les 2 cas suivants:

- le cas où il existe 2 ou plusieurs routes à connexions comme seules alternatives pour se rendre de p à q .
- le cas où il existe 1 ou plusieurs routes à connexions et où il existe d'autres routes pour se rendre de p à q .

On a rejeté l'approche qui consiste à considérer toutes les routes à connexions à cause du trop grand nombre de ces routes qui seraient générées. Ainsi, lorsqu'il n'y a qu'une route à connexion pour se rendre de p à q , on répartit ce marché sur tous les marchés correspondant aux seg-

ments de la route à connexions. On accepte donc une faible erreur sur le coût du billet venant du fait que la somme des coûts des billets sur tous les segments est supérieure au prix du billet pour aller de p à q .

Ainsi, par exemple, si l'on n'a que les 2 routes suivantes:

1 - a-c-f

2 - e-c-d

FIGURE 6 ROUTE A CONNEXIONS UNIQUE

Un passager voulant se rendre de a à d fera sûrement une connexion à c ; mais une telle connexion ne nous intéresse pas. Il doit être clair que ce passager fait partie dans ce cas de 2 marchés: soit le marché a-c et le marché c-d.

A ce stade, il convient de réintroduire la notation en insistant sur les modifications apportées dans le contexte des connexions.

3.1 Définition des ensembles

Pour les routes, rappelons que nous avons les ensembles suivants:

- R = {routes du réseau}
- R_{pqm} = {routes desservant le marché pq en m arrêts}
- $R_{pq} = \bigcup_{m} R_{pqm}$

Pour les routes à connexions, nous avons les ensembles suivants:

- RC = {routes à connexions du réseau}
- C_{pq} = {routes à connexions où l'on retrouve le tronçon pq}

  $\{ c \mid (p,q) \in C_c \}$
- C_c = {tronçons de routes formant la route à connexion c}

  $\{ (p_i, p_{i+1}) \}$ $\quad 1 \leq i \leq n$
- RC_{pqs} = {routes à connexions avec départ à p et arrivée à q en s tronçons}
- $RC_{pq} = \bigcup_{s} RC_{pqs}$

3.2 Modifications du modèle lui-même

3.2.1 Définition des nouveaux symboles et des approximations

c : 1 route à connexion

s : l'indice désignant le nombre de tronçons de routes réelles utilisés par une route à connexions

P_c : nombre de passagers sur c

PT_{pqr} : nombre de passagers utilisant la route r entre p et q comme tronçon de route à connexions

K_c : coefficient de fréquence pour une route à connexion. Si "min" est défini comme le minimum de services pris sur tous les tronçons, si "$\sum_s$ ser" est défini comme la somme des services pris sur tous les tronçons et si R_s est défini comme un coefficient de pénalité sur s tel que $R_2 = K_2$ et $R_3 = K_4$,

(vg: K_m tel que défini à la section 2.2.3)

alors:

$$K_c = \frac{\min}{\sum_s ser} \cdot R_s$$

L'adjonction de ce coefficient provient du fait que l'équivalent-service d'une route à connexions se rapproche du minimum de services pris sur tous les tronçons. (vg. contraintes de type 3).

F_{pq} , FD_{pq} , T_{pq} , TD_{pq} : ces termes sont illustrés sur la figure 7.

La courbe de fréquence est restreinte à 2 segments de droite telle qu'illustrée sur la figure 7 : le premier de longueur finie et l'autre théoriquement de longueur infinie.

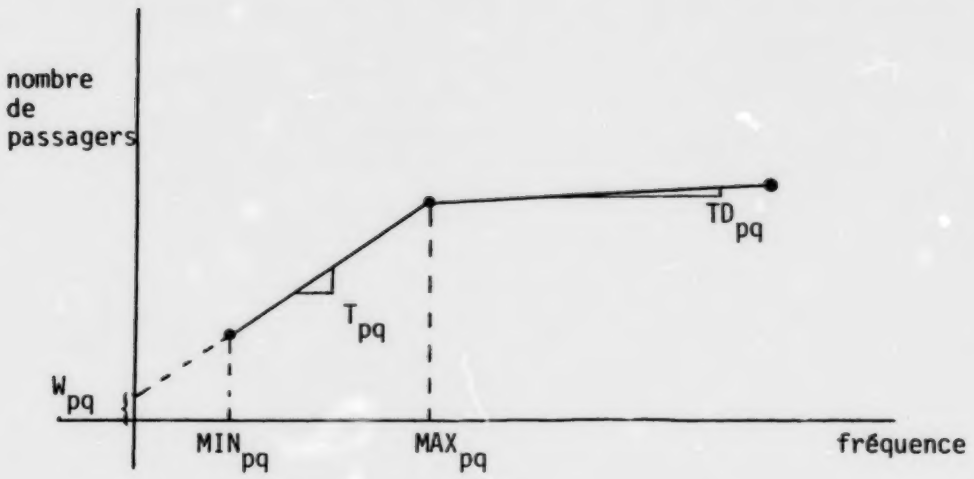


FIGURE 7 COURBE SEGMENTEE DE FREQUENCE VS DEMANDE DANS LE MODELE MODIFIE

Dans un premier temps, pour chaque marché p-q , on calcule la fréquence équivalente actuelle du service en utilisant une contrainte de type 3 du modèle modifié. On obtient MIN_{pq} et MAX_{pq} en prenant cette fréquence équivalente actuelle respectivement diminuée et augmentée d'un certain pourcentage sur lequel il y a lieu de paramétriser. Ces deux quantités constituent les bornes d'un intervalle. Associons la variable F_{pq} à cet intervalle et alors

$$MIN_{pq} \leq F_{pq} \leq MAX_{pq}$$

Si on dénote par T_{pq} la pente du premier segment de droite, alors T_{pq} est calculé de la façon suivante:

$$T_{pq} = \frac{\text{demande actuelle}}{\text{fréquence équivalente actuelle}} \cdot Y$$

où Y a été ajouté pour tenir compte du fait que la pente ne correspond pas directement au quotient de la demande actuelle sur la fréquence équiva-

lente actuelle. (i.e. pour tenir compte du fait que le prolongement à gauche du premier tronçon de la figure 7 ne passe pas par l'origine). Y permet de représenter la sensibilité relative de la demande à une faible variation de fréquence. Y sera également paramétrisé.

Finalement on a défini la pente du deuxième segment de droite, TD_{pq} , comme il suit:

$$TD_{pq} = 0.1 \quad T_{pq} \qquad \text{(il y a lieu de paramétriser là aussi)}$$

Ceci reflète le fait qu'au delà de MAX_{pq} , la fréquence équivalente exerce une influence beaucoup moindre sur la demande comparativement à celle exercée dans l'intervalle $[MIN_{pq} , MAX_{pq}]$. A ce niveau on considère la demande comme pratiquement saturée. La variable associée à ce second segment est dénotée par FD_{pq}.

Il est à noter que W_{pq} représente l'ordonnée à l'origine du premier segment de droite; W_{pq} apparaîtra dans les contraintes de type 4.

3.2.2 Redéfinition du modèle

La fonction économique est légèrement modifiée pour tenir compte des revenus supplémentaires des passagers sur les routes à connexions:

$$MAX \left\{ \sum_{pq} Y_{pq} \left(\sum_{r \in R_{pq}} P_{pqr} + \sum_{c \in RC_{pq}} P_c \right) - \sum_{r} \sum_{a} DC_{ra} \cdot N_{ra} \right\}$$

- LES CONTRAINTES DE TYPE 1 :

Le nombre total de passagers utilisant le segment i-j sur la route r doit être plus petit ou égal au nombre total de sièges disponibles

sur cette route r .

$$\sum_{a} LF_{i-j} \cdot S_a \cdot N_{ra} - \sum_{\substack{p<j \\ i<q}} P_{pqr} - \sum_{\substack{p<j \\ i<q \\ C_{pq}\neq\phi}} PT_{pqr} \geq 0 \qquad \forall (i-j) , \forall r$$

- LES CONTRAINTES DE TYPE 2 :

Le nombre total de passagers utilisant les routes à connexions qui ont p-q comme tronçon est distribué sur les routes normales qui desservent le marché p-q .

$$\sum_{c\in C_{pq}} P_c - \sum_{r\in R_{pq}} PT_{pqr} = 0$$

$$\forall (p,q) \Rightarrow C_{pq} \neq \phi$$

- LES CONTRAINTES DE TYPE 3 :

La fréquence hebdomadaire est égale à la somme des équivalents de fréquence sur les routes plus la somme des équivalents de fréquence sur les routes à connexions.

$$\sum_{m} K_m \sum_{a} K_a \sum_{r\in R_{pqm}} N_{ra} +$$

$$\sum_{s} \left(\sum_{c\in RC_{pqs}} 1 \sum_{p_i,p_{i+1}\in C_c} \sum_{m} K_m \sum_{a} K_a \sum_{r\in R_{p_i p_{i+1} m}} N_{ra} \right] \cdot K_c)$$

$$- F_{pq} - FD_{pq} = 0 \quad , \forall (p,q)$$

- LES CONTRAINTES DE TYPE 4 :

Le nombre total de passagers allant de p à q est fonction de la fréquence de services. (vg. p. 21)

$$W_{pq} + F_{pq} \cdot T_{pq} + FD_{pq} \cdot TD_{pq} - \sum_{r \in R_{pq}} P_{pqr} - \sum_{c \in RC_{pq}} P_c \geq 0 \quad , \forall (p,q)$$

- LES CONTRAINTES DE TYPE 5 :

La disponibilité de la flotte d'avions:

$$\sum_r TB_{ra} \cdot N_{ra} \leq U_a \cdot A_a \quad , \forall a$$

où TB_{ra} = temps nécessaire pour parcourir la route r avec un avion de type a

U_a = nombre d'heures d'utilisation disponibles par semaine pour un avion de type a

A_a = nombre d'avions de type a disponibles

- LES CONTRAINTES DE TYPE 6 : l'imposition des bornes sur F_{pq}

$$MIN_{pq} \leq F_{pq} \leq MAX_{pq} \quad , \forall (p,q)$$

NOTE: pour l'instant, on laisse tomber les contraintes de type 6 du modèle original (vg. le nombre maximum de départ par ville) .

3.3 Commentaires généraux

3.3.1 Les données:

Il y a 2 types de données qui se répartissent comme suit: les données brutes et les données générées par le système à partir des données brutes.

Les données brutes sont:

1) Y_{pq} : les coûts des billets

2) DC_{ra} : les coûts directs d'opération

3) LF_{i-j} : les coefficients de remplissage

4) S_a : le nombre de sièges par type d'avion

5) TB_{ra} : le temps nécessaire pour parcourir la route r par un avion de type a

6) U_a : le nombre d'heures d'utilisation disponibles par semaine pour un avion de type a

7) A_a : le nombre d'avions de type a disponible

8) R et RC : les 2 ensembles de bases

9) D_{pq} : la demande hebdomadaire pour chaque marché

10) N_{ra} actuels : c'est-à-dire la fréquence actuelle des vols

11) K_m, K_a : les coefficients de fréquence associés aux arrêts et au type d'avions

12) m, a, s : le nombre d'arrêts maximum sur une route, les types d'avions, le nombre maximum de tronçons sur une route à connexions.

Les données générées par le système sont:

1) les ensembles suivants:

R_{ij} , R_{pq} , R_{pqm} , RC_{pq} , RC_{pqs} , C_c , C_{pq}

2) MIN_{pq} , MAX_{pq} , T_{pq} , TD_{pq}, W_{pq} générées à partir de 9) et 10) des données brutes et des contraintes de type 3

3) K_c : les coefficients de fréquence pour les routes à connexions.

3.3.2 Les variables:

Il y a 6 types de variables:

1) P_{pqr}
2) P_c
3) PT_{pqr}
4) N_{ra}
5) F_{pq}
6) FD_{pq}

3.3.3 Le nombre de contraintes et de variables

On en déduit le nombre de contraintes et de variables de la façon suivante: si on définit

| pq | = le nombre de couples origine-destination dans le réseau

| a | = le nombre de types d'avions

| { } | = la cardinalité d'un ensemble

alors le modèle modifié comporte

un nombre de contraintes égale à $NC \leq 4 \mid pq \mid + \mid R_{ij} \mid + \mid a \mid$

et un nombre de variables égale à $NV \leq \mid R \mid . \mid a \mid + 2 \mid pq \mid + 2 \mid R_{pq} \mid + \mid RC_{pq} \mid$

comparativement au modèle original qui comportait

un nombre de contraintes égale à $NC \leq 3 \mid pq \mid + \mid R_{ij} \mid + \mid a \mid + \mid Q \mid$

et un nombre de variables égale à $NV \leq \mid R \mid . \mid a \mid + 3 \mid pq \mid + \mid R_{pq} \mid$

Notons que le nombre de $4 \mid pq \mid$ dans le calcul du nombre de contraintes sera effectivement de $3 \mid pq \mid$ au niveau de APEX-II puisque les contraintes de type 6 seront introduites comme bornes sur les variables et non comme contraintes explicites. Le $\mid Q \mid$ n'apparait pas dans le calcul du nombre de contraintes du modèle modifié puisqu'on ne se préoccupe pas des contraintes de type 6 qui apparaissent dans le modèle original.

Au niveau des variables, on est passé de $3 \mid pq \mid$ à $2 \mid pq \mid$ puisque le nombre de segments linéaires de la courbe d'approximation de la demande en fonction de la fréquence est passé de 3 à 2. Finalement l'adjonction des PT_{pqr} se traduit par une augmentation de $\mid R_{pq} \mid$ variables et l'adjonction des P_c par une augmentation de $\mid RC_{pq} \mid$ variables.

Théoriquement, il en découle que ce modèle modifié tiendra compte des connexions tout en ne perdant pas de vue les objectifs du modèle original qui sont:

1) de déterminer des fréquences sur des routes pour maximiser le profit
2) de choisir les types d'avions qui conviennent le mieux.

Bien sûr, des études de paramétrisation seront à faire à la suite de ce stage mais allons voir maintenant comment on a procédé pour l'implantation du fichier d'entrée pour APEX-II.

CHAPITRE IV

TECHNIQUE D'IMPLANTATION ET DOCUMENTATION

La résolution du problème de programmation linéaire se fera à l'aide de APEX-II. Il faut donc écrire le programme de la génération des contraintes dans un format acceptable par APEX-II.

Or, pour générer les contraintes, il faut s'assurer de générer les bons ensembles auxquels se réfèrent les contraintes; et pour générer ces bons ensembles, il faut se doter d'une bonne structure de représentation des routes et des routes à connexions.

4.1 Représentation des routes et des routes à connexions

Le choix s'est arrêté sur une structure de listes. Cette représentation s'effectue dans la sous-routine STRUCM. Au début, on dispose de la liste des villes du réseau qu'on place, triée, dans le vecteur VILLE(I); le vecteur IPT(I) est associé à VILLE(I) et pointe sur la dernière route rencontrée qui dessert VILLE(I).

X(I) et IX(I) sont des vecteurs qui recoivent respectivement:

X(I) : les numéros de routes et les villes qui composent les routes

IX(I) : le nombre de villes pour une route donné et, associé à chaque ville de X(I), un pointeur à l'avant-dernière route rencontrée desservant VILLE(I). Lorsque ce pointeur est nul, ceci indique qu'on est en présence de la première route desservant VILLE(I).

On utilise l'exemple suivant pour illustrer:

Supposons qu'on ait 4 routes impliquant 5 villes :

RR0001 : YYZ - YUL

RR0002 : YYZ - YQB

RR0003 : YYZ - YUL - YQB

RR0004 : YWG - YYZ - YQB - YHZ

Initialement on a :

VILLE	IPT
YHZ	0
YQB	0
YUL	0
YWG	0
YYZ	0

Le vecteur VILLE est trié et IPT est à 0. Les vecteurs X et IX sont vides.

FIGURE 8 REPRESENTATION DES VECTEURS VILLE, IPT, X et IX

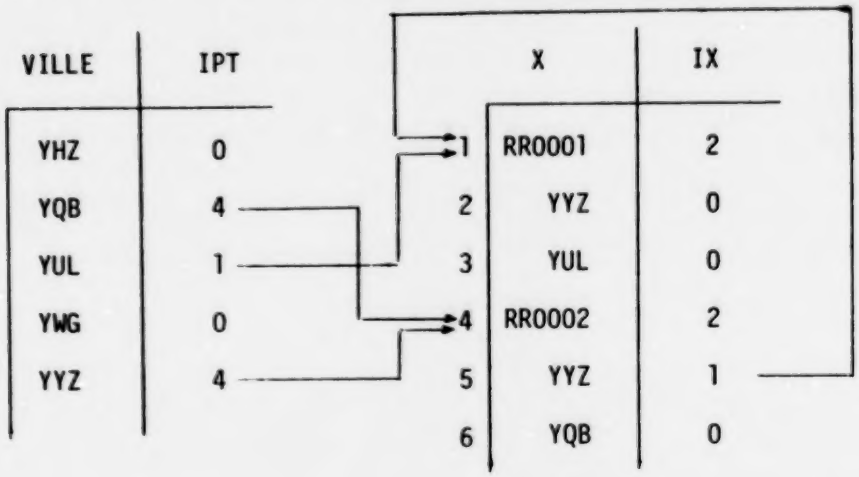
Après avoir considéré la première route, on retrouve

VILLE	IPT
YHZ	0
YQB	0
YUL	1
YWG	0
YYZ	1

	X	IX
1	RR001	2
2	YYZ	0
3	YUL	0

FIGURE 9 REPRESENTATION DES VECTEURS VILLE, IPT, X et IX

Après avoir considéré les deux premières routes, on retrouve:



VILLE	IPT
YHZ	0
YQB	4
YUL	1
YWG	0
YYZ	4

	X	IX
1	RR0001	2
2	YYZ	0
3	YUL	0
4	RR0002	2
5	YYZ	1
6	YQB	0

FIGURE 10 REPRESENTATION DES VECTEURS VILLE, IPT, X et IX

YYZ est desservie par les deux premières routes RR001 et RR0002. Alors le IPT associé à YYZ pointe vers la deuxième route et le IX associé à YYZ dans la description de la deuxième route, pointe vers la première route.

Après avoir considéré les trois premières routes, on retrouve:

VILLE	IPT
YHZ	0
YQB	7
YUL	7
YWG	0
YYZ	7

	X	IX
1	RR0001	2
2	YYZ	0
3	YUL	0
4	RR0002	2
5	YYZ	1
6	YQB	0
7	RR0003	3
8	YYZ	4
9	YUL	1
10	YQB	4

FIGURE 11 REPRESENTATION DES VECTEURS VILLE, IPT, X et IX

Finalement on a:

VILLE	IPT
YHZ	11
YQB	11
YUL	7
YWG	11
YYZ	11

	X	IX
1	RR0001	2
2	YYZ	0
3	YUL	0
4	RR0002	2
5	YYZ	1
6	YQB	0
7	RR0003	3
8	YYZ	4
9	YUL	1
10	YQB	4
11	RR0004	4
12	YWG	0
13	YYZ	7
14	YQB	7
15	YHZ	0

FIGURE 12 REPRESENTATION DES VECTEURS VILLE, IPT, X et IX

STRUCM génère cette structure pour toutes les routes du réseau. En l'appellant une seconde fois, on génère cette même structure pour les routes à connexions de la façon suivante:

a) IPC remplace IPT et est mis à 0.

b) Les routes à connexions s'empilent à la suite des routes dans les vecteurs X et IX .

4.2 Génération des ensembles

Avant d'écrire les sous-routines définies subséquemment, on a cherché à se servir du sous-ensemble MRG (Matrix and Report generator) de APEX-II. MRG offrait, entre autres, la possibilité de définir des listes de base et la possibilité de générer des listes particulières en partant des listes de bases. Cependant le nombre de listes permises était borné à 100, ce qui était insuffisant pour ce cas.

Ainsi donc, 3 sous-routines de base vont générer les ensembles:

1- GENRPQ(P,Q) génère R_{pq}

2- GERCPQ(P,Q,IPC) génère RC_{pq}

3- GENCPQ(P,Q,IPC) génère C_{pq}

Deux autres sous-routines GECRPQ(P,Q,NARR) et GCRCPQ(P,Q,IPC) accomplissent respectivement un travail similaire à celui de GENRPQ et GERCPQ mais en plus:

- génèrent les ensembles R_{pqm} et RC_{pqs}
- génèrent des variables selon le format de APEX-II.

Ces dernières se sont imposées à cause des difficultés posées par les contraintes de type 3 où il faut déterminer un coefficient K_c pour chaque route à connexions et où il faut déterminer un équivalent fréquence actuel (à l'aide des N_{ra} actuels) pour borner les F_{pq}. De plus, il faut déterminer à ce niveau les T_{pq} et TD_{pq} qui reviendront comme coefficients dans les contraintes de type 4.

Le détail des 5 sous-routines sera présenté d'une façon algorithmique.

4.2.1 GENRPQ(P,Q)

Cette sous-routine trouve toutes les routes qui desservent le marché p-q .

L'algorithme est le suivant:

a) On appelle TROUVE pour trouver le pointeur correspondant à P . Ce pointeur s'appelle K et pointe sur le numéro de route où la ville P a été rencontrée pour la dernière fois.

b) L devient le nombre de villes associées à cette route.

c) On parcourt séquentiellement toutes les villes de cette route à la recherche de P . Lorsqu'on a trouvé P, on dénote par J l'indice de P . Si P est la dernière ville de la route on va à d) ; sinon on va à f).

d) Si P est la dernière ville, alors on examine son pointeur.

e) Si ce pointeur est nul, alors c'est la fin. Sinon, K devient ce nouveau pointeur et on retourne à b).

f) Si P n'est pas la dernière ville, on cherche dans les villes qui restent, la ville Q .

g) Si Q n'y est pas, alors on examine le pointeur de P et on retourne à e).

h) Si Q y est, alors RPQ(NN) reçoit cette route et IRPQ(NN) reçoit le nombre d'arrêt (m) entre P et Q . Puis on examine le pointeur de P et retour à e).

N.B.: A la fin, NN représente le nombre de routes trouvées.

4.2.2 GERCPQ(P,Q,IPT)

Cette sous-routine trouve toutes les routes à connexions qui commencent à P et qui se terminent à Q .

NOTE: A l'appel, IPC remplace le paramètre formel IPT .

L'algorithme est le suivant:

a) On appelle TROUVE pour trouver le pointeur correspondant à P . Ce pointeur s'appelle K et pointe sur le numéro de route à connexions où la ville P a été rencontrée pour la dernière fois.

b) L devient le nombre de villes associées à cette route à connexions.

c) Si P est la première ville de cette route et Q est la dernière ville de cette route, alors on retient dans les vecteurs RPQ(NN) et IRPQ(NN), la route et le nombre de connexions. Puis on retourne à e)

d) Sinon, on cherche l'occurrence de P .

e) Si le pointeur de P est nul, alors c'est la fin. Sinon, K devient ce nouveau pointeur et on retourne à b).

NOTE: NN représente toujours le nombre de routes trouvées.

4.2.3 GENCPQ(P,Q,IPT)

Appelée dans la contrainte 2, cette sous-routine trouve toutes les routes à connexions qui ont pour tronçons le marché p-q . Appelée dans la contrainte 1, elle trouve toutes les routes normales qui ont pour tronçon le marché p-q.

L'algorithme est le suivant:

a) On appelle TROUVE pour trouver le pointeur correspondant à P. Ce pointeur s'appelle K et pointe sur le numéro de route (ou de route à connexions) où la ville P a été rencontrée pour la dernière fois.

b) L devient le nombre de villes associées à cette route (ou route à connexions).

c) On parcourt séquentiellement toutes les villes de cette route à la recherche de P . Lorsqu'on a trouvé P , on dénote son indice par J . Si P est la dernière ville de la route on va a d); sinon, on va à f).

d) Si P est la dernière ville, alors on examine son pointeur.

e) Si ce pointeur est nul, c'est la fin. Sinon, K devient ce nouveau pointeur et on retourne à b).

f) Si P n'est pas la dernière ville, alors on vérifie si la ville suivante est Q . Sinon, on examine le pointeur de P et on retourne à e).

g) Si Q est la ville suivante, alors RPQ(NN) reçoit cette route (ou route à connexions) et BVILL(NN,1) et BVILL(NN,2) reçoivent respectivement la première et la dernière ville de cette route.

NOTE: NN représente le nombre de routes trouvées.

4.2.4 GECRPQ(P,Q,NARR)

La première partie de cette sous-routine est analogue à GENRPQ sauf que lorsqu'on a trouvé une route et qu'on l'assigne au vecteur RPQ, on conserve aussi l'indice de cette route dans le vecteur IND; ce dont on aura évidemment besoin en 2^{o} partie.

Les objectifs de la 2^{o} partie sont de trouver les R_{pqm} , d'écrire les variables sous le format de APEX-II et de calculer des fréquences équivalentes.

Notons que NARR représente le nombre d'arrêts maximum sur une route et permettra de paramétriser cette partie des contraintes de type 3 au niveau du m :

$$\sum_{(p_i,p_{i+1})\in C_c} \sum_{m} K_m \sum_{r\in R_{p_i p_{i+1} m}} N_{ra}$$

Ainsi, puisque pour l'instant on ne s'occupe que des services directs à mettre sur les connexions, cette partie devient:

$$\sum_{(p_i,p_{i+1})\in C_c} \sum_{r\in R_{p_i p_{i+1} 1}} N_{ra}$$

Notons que ASPQ est la variable qui recevra la fréquence équivalente, éventuellement.

L'algorithme est le suivant:

a) On appelle TROUVE pour trouver le pointeur correspondant à P . Ce pointeur s'appelle K et pointe sur le numéro de route où la ville P a été rencontrée pour la dernière fois.

b) L devient le nombre de villes associées à cette route.

c) On parcourt séquentiellement toutes les villes de cette route à la recherche de P . Lorsqu'on a trouvé P , on dénote son indice par J . Si P est la dernière ville de la route, on va à d); sinon, on va à f).

d) Si P est la dernière ville, alors on examine son pointeur.

e) Si ce pointeur est nul, alors on va à i). Sinon, K devient ce nouveau pointeur et on retourne à b).

f) Si P n'est pas la dernière ville, on cherche dans les villes qui restent, la ville Q .

g) Si Q n'y est pas, alors on examine le pointeur de P et on retourne à e).

h) Si Q y est, alors RPQ(NN) reçoit cette route et IRPQ(NN) reçoit le nombre d'arrêt (m) entre P et Q . On conserve également K dans IND(NN). Puis on examine le pointeur de P et on retourne à e).

i) ASPQ est mis à zéro. On classifie les routes trouvées selon le nombre d'arrêt (m) (R_{pqm}).

j) Puis, $\forall R_{pqm}$, $1 \leq m \leq 6$, on vérifie si ce sous-ensemble est vide. Si oui, on passe à la valeur suivante de m ; sinon, on appelle la fonction AKM qui retourne le K_m exigé.

k) Pour tout $r \in R_{pqm}$, on boucle sur les types d'avions en écrivant la variable N_{ra} correspondante. Après un appel à TROUVE, on calcule la fréquence équivalente basée sur les fréquences actuelles; ces fréquences actuelles se retrouvent dans IROUT(J). Cependant, si NARR = 1 , alors on ne s'occupe que des services directs; ce cas se produit dans la sous-routine GCRCPQ qui appelle GECRPQ(P,Q,1).

4.2.5 GCRCPQ(P,Q,IPT)

La première partie de cette sous-routine est analogue à GERCPQ sauf que lorsqu'on a trouvé une route et qu'on l'assigne au vecteur RPQ, on conserve aussi l'indice de cette route dans le vecteur IND. De plus GCRCPQ ne s'engage pas automatiquement dans sa 2^{o} partie; s'il ne trouve pas de route dans la 1^{re} partie, ASPQ est mis à 0 et l'on sort.

Les objectifs de la 2^{o} partie sont de trouver les RC_{pqs} , d'écrire des variables sous le format de APEX-II, par le biais des appels à GECRPQ, et de calculer des équivalents-fréquence.

IPT est le paramètre formel qui est remplacé par IPC à l'appel.

L'algorithme est le suivant:

a) On appelle TROUVE pour trouver le pointeur correspondant à P . Ce pointeur s'appelle K et pointe sur le numéro de route à connexions où la ville P a été rencontrée pour la dernière fois.

b) L devient le nombre de villes associées à cette route à connexions.

c) Si P égale la 1^{re} ville de cette route et Q égale la dernière ville de cette route, alors on retient dans les vecteurs RPQ et IRPQ la route et le nombre de connexions. De même, on conserve K dans IND(NN). Puis, on va à e).

d) Sinon, on cherche l'occurrence de P .

e) Si le pointeur de P est nul, on va à f). Sinon, K devient ce nouveau pointeur et retour à b).

f) Si NN = 0 , alors ASPQ est mis à zéro et on sort.

g) Si NN ≠ 0 , on classifie les routes trouvées selon le nombre de connexions (RC_{pqs}) .

h) Puis, pour chaque p-q qui détermine un tronçon et pour chaque sous-ensemble déterminé en g) , on appelle GECRPQ(P,Q,1). Parallèlement, on détermine le K_c .

4.2.6 TROUVE et AKM

La sous-routine TROUVE effectue une recherche binaire d'un élément dans un vecteur évidemment trié au préalable. On sauve ainsi beaucoup de temps puisque la recherche d'un élément dans une table se présente fréquemment.

On a rédigé aussi une fonction appelée AKM qui retourne la valeur K_m en fonction de m , le nombre d'arrêts sur une route (+1).

4.3 Génération des contraintes

Six sous-routines génèrent les contraintes, dans le cadre de APEX-II, en se servant des routines précédentes. Pour chaque ligne de la matrice, le LGL et les AIJ détaillés sont générés. (Le LGL sert à dénommer chaque contrainte tandis que le AIJ explicite la variable). Quand il le faut, les RHS et les BND le sont aussi. (RHS pour Right Hand Side et BND pour Bound).

1- FONCOB(IPC) : génère la fonction objective

2- CONTR1(IPC) : génère les équations de type 1

3- CONTR2(IPC) : génère les équations de type 2

4- CONTR3(IPC) : génère les équations de type 3

5- CONTR4(IPC) : génère les équations de type 4

6- CONTR5 : génère les équations de type 5

4.3.1 FONCOB(IPC)

Cette sous-routine génère la fonction objective.

L'algorithme est le suivant:

a) On calcule les ADC(R,A), c'est-à-dire les coûts directs d'opération.

b) On écrit le LGL .

c) On parcourt tous les marchés P-Q et pour chacun d'eux:

i) Chaque variable P_{pqr} générée par GENRPQ(P,Q) est affectée d'un coefficient égal à Y(PQ) , le coût du billet pour aller de P à Q.

ii) On appelle GERCPQ(P,Q,IPC) et si le nombre de routes à connexions est égal à 0, on passe à l'analyse du prochain marché.

iii) Sinon, chaque variable P_c générée par GERCPQ(P,Q,IPC) est affectée d'un coefficient égal à Y(PQ).

d) On parcourt toutes les routes et pour chacune d'elles, on parcourt tous les types d'avions afin de générer toutes les variables N_{ra} qui sont affectées des coefficients - ADC(R,A).

4.3.2 CONTR1(IPC)

On doit générer une contrainte de ce type pour chaque segment de chaque route. Pour percevoir la signification de $\sum_{\substack{p<j \\ i\leq q}} P_{pqr}$, analysons l'exemple suivant:

Considérons la route

Cette route est constituée de 3 segments:A-B, B-C et C-D. Il faut donc générer 3 contraintes. Considérant le segment A-B , il faut regarder tous les marchés qui pourraient emprunter ce segment sur cette route. Il pourrait y avoir trois marchés au maximum: A-B , A-C et A-D . De même pour le segment B-C , il pourrait y avoir les marchés suivants: A-C , B-C , B-D et A-D . Finalement pour le segment C-D , les marchés à considérer sont A-D , B-D et C-D . Or ces marchés ne sont pas nécessairement tous définis ou encore tous desservis par une compagnie d'aviaton. Il faut donc envisager toutes les possibilités de marché et pour chaque marché, aller vérifier son occurrence dans une table des marchés définis.

Ainsi pour un segment (i,j) , la liste des marchés possibles à générer sont tous ceux qui commencent avant ou sur i et qui finissent sur ou après j .

L'algorithme est le suivant:

a) On génère un vecteur RPETQ qui contiendra tous les marchés P-Q en format R6. (Les 6 lettres formées par P et Q).

b) On lit le nombre de sièges par avion, les segments des routes et leur coefficient de remplissage. Pour toutes les routes et pour chaque (i,j) sur une route, on écrit son LGL et son RHS .

c) Puis, en bouclant sur les types d'avions, on écrit les variables N_{ra} avec $(LF_{i-j} \cdot S_a)$ comme coefficient.

d) Pour tous les marchés qui existent et qui commencent avant ou sur i et qui finissent sur ou après j on écrit la variable P_{pqr} avec -1 comme coefficient.

e) De plus, si l'ensemble C_{pq} n'est pas vide, alors on écrit la variable PT_{pqr} avec -1 comme coefficient.

4.3.3 CONTR2(IPC)

On a une telle contrainte pour chaque marché p-q .

L'algorithme est le suivant:

a) On appelle GENCPQ qui retournent les NN routes à connexions trouvées.

b) Si NN = 0 , on passe au prochain marché. Sinon, on écrit le LGL et le RHS .

c) Puis on écrit les variables P_{pqc} (ou les P_c) avec un coefficient de +1 .

d) Ensuite on écrit les variables PT_{pqr} , où les r ont été fournies par GENRPQ, avec un coefficient de -1 .

4.3.4 CONTR3(IPC)

On a une telle contrainte pour chaque marché p-q.

L'algorithme est le suivant:

a) On lit ACF et APT qui représente respectivement le pourcentage d'écart pour déterminer l'intervalle de variation de F_{pq} et le coefficient Y dans la définition de la pente T_{pq} .

b) On écrit le LGL et le RHS .

c) On appelle GECRPQ et GCRCPQ qui permettent de calculer le ASPQ duquel on déduit les bornes de l'intervalle de variation pour F_{pq} et la pente du segment. Il ne faut pas oublier que les 2 routines écrivent les N_{ra} .

d) Puis on écrit les variables F_{pq} et FD_{pq} avec un coefficient de -1 .

e) Finalement, on écrit les BND déterminant l'intervalle de variation de F_{pq} .

4.3.5 CONTR4(IPC)

On a également une telle contrainte pour chaque marché p-q .

L'algorithme est le suivant:

a) On écrit le LGL , le RHS et la variable F_{pq} avec T_{pq} comme coefficient (a été calculé dans CONTR3).

b) On écrit FD_{pq} avec TD_{pq} comme pente. TD_{pq} est sujet à une paramétrisation.

c) Puis, pour les NN routes fournies par GENRPQ , on écrit les variables P_{pqr} correspondantes avec un coefficient de -1 .

d) Puis, pour les NN routes fournies par GERCPQ , on écrit les variables P_{pqc} (ou les P_c) correspondantes avec un coefficient de -1 .

4.3.6 CONTR5

On a une telle contrainte pour chaque type d'avions.

L'algorithme est le suivant.

a) On lit U_a (AU(I) dans le programme) qui représente le nombre d'heures d'utilisation disponibles par semaine pour chaque type d'avions et A_a(WAA(I) dans le programme) qui représente le nombre d'avions de chaque type disponibles.

b) Pour chaque type d'avions, on écrit le LGL et le RHS .

c) Puis, pour chaque route, on écrit la variable N_{ra} avec Tb_{ra} (TVOL(I) dans le programme) comme coefficient.

4.4 Programme principal

Finalement le programme principal GENCON ne fait que les lectures de:

- NV : nombre de villes dans le réseau
- NPQ : nombre de couples origine-destination
- NAV : nombre de types d'avions
- NNIJ : nombre de segments (I-S)
- VILLE(I) : les villes du réseau
- Y(K) : le coût d'un billet/marché
- OANDD(K) : la demande d'un marché
- IP(K),IQ(K) : les P et les Q
- TYPAV(J) : les types d'avions

Puis, il fait 2 appels à STRUCM, écrit 'FILE' sur le fichier qu'on veut constituer, appelle dans l'ordre FONCOB, CONTR1, CONTR2, CONTR3, CONTR4, CONTR5 et écrit 'ENDFILE' complétant ainsi le fichier d'entrée pour APEX-II.

La figure 13 indique les relations entre le programme principal et entre les sous-routines. Il faut noter qu'une flèche indique 1 ou plusieurs appels à la sous-routine impliquée.

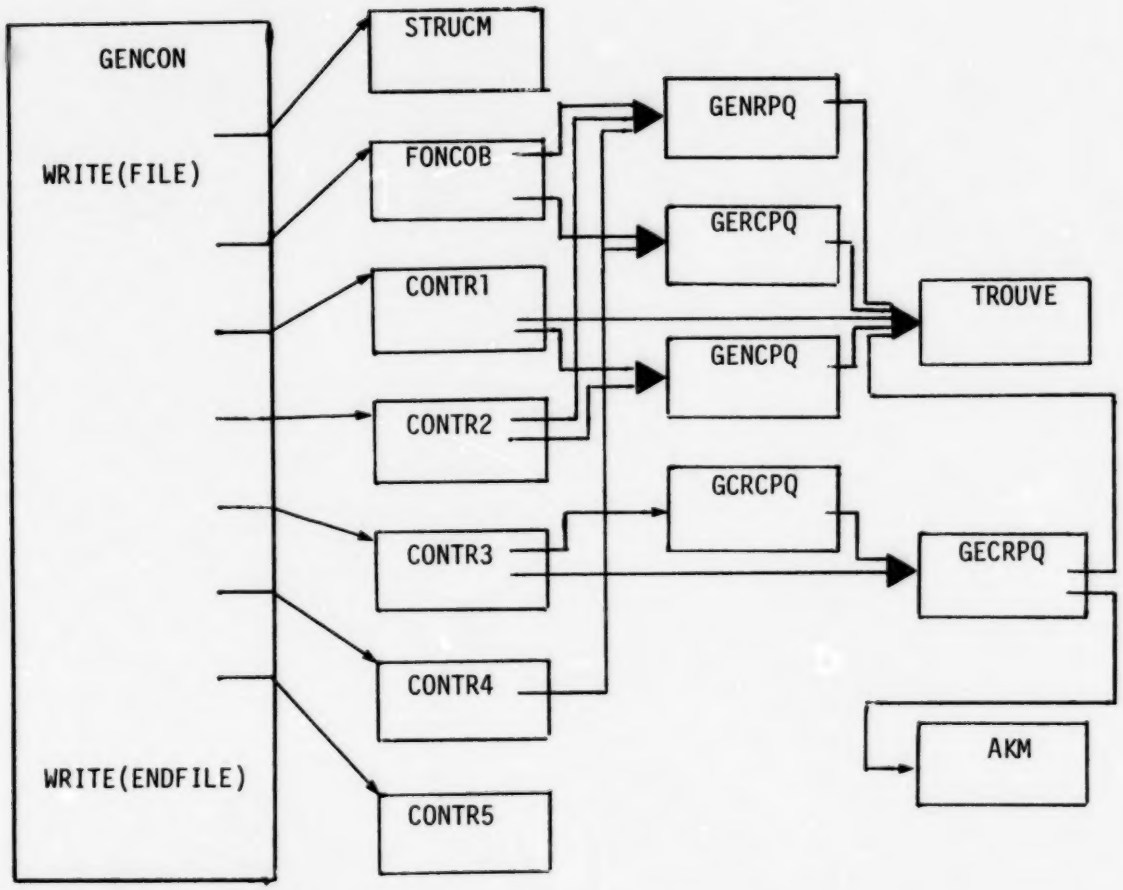


FIGURE 13 RELATIONS ENTRE LE PROGRAMME PRINCIPAL
ET SES ROUTINES

REMERCIEMENTS

Je tiens à remercier Monsieur Jean-Marc Rousseau et Monsieur Jacques Ferland pour leur appui continu au niveau de la conception et de l'analyse. De plus, j'aimerais remercier Monsieur Réjean Lessard ainsi que Monsieur Guy Lapalme pour leur aide précieuse au niveau de la programmation. Je suis reconnaissant à Madame Jacqueline Gariépy pour son excellent travail de dactylographie.

BIBLIOGRAPHIE

[1] William Swan, THE COMPLEAT FA-4 MEMO, FTL Technical Memorandum 72-10, Flight Transportation Laboratory Department of Aeronautics and Astronautics, Massuchusetts Institute of Technology, August 1972.

[2] APEX-II REFERENCE MANUAL, Control Data Corporation, Publication No. 5915800, Minneapolis, Minnesota, 1974.

ANNEXE

LE PROGRAMME ET SES ROUTINES

```
      PROGRAM GENCON(INPUT,OUTPUT,TAPE1)
      IMPLICIT INTEGER (B-Z)
      COMMON /V1/ VILLE(100),X(5000),IX(5000),TAM(7)
      COMMON /V2/ RVILL(100,2),RPQ(100),IRPQ(100),TEMP(100)
      COMMON /V3/ IP(1000),IQ(1000),QANDQ(1000),Y(1000)
      COMMON /V4/ NV,NSPQ,NPQ,NNIJ
      COMMON /V5/ IPT(100),ROUT(500),IROUT(500)
      COMMON /V6/ NBROUT,NAV,TYPAV(8)
      COMMON /V7/ NN
      COMMON /V8/ ATPQ(1000)
      COMMON /V9/ P1,Q1
      COMMON /V10/ TVOL(500)
      DIMENSION CROUT(500),CIROUT(500),IPC(100)
      READ 100,NV,NPQ,NAV,NNIJ
  100 FORMAT(4I4)
      READ 101,(VILLE(J),J=1,NV)
  101 FORMAT(20(1X,R3))
      READ 102,(Y(K),QANDQ(K),IP(K),IQ(K),K=1,NPQ)
  102 FORMAT(2I6,1X,R3,1X,R3)
      READ 104,(TYPAV(J),J=1,NAV)
  104 FORMAT(8R4)
      I=1
      NCROUT=0
      NBROUT=0
      CALL STRUCM(I,IPT,ROUT,IROUT,NOROUT,NBROUT)
      NCROUT=0
      CALL STRUCM(I,IPC,CROUT,CIROUT,NOROUT,NCROUT)
      WRITE(1,103)
  103 FORMAT(' FILE',T11,'TAPE1')
      CALL FONCOB(IPC)
      CALL CONTR1(IPC)
      CALL CONTR2(IPC)
      CALL CONTR3(IPC)
      CALL CONTR4(IPC)
      CALL CONTR5
      WRITE(1,105)
  105 FORMAT(' ENDFILE')
      STOP
      END
```

```
      SUBROUTINE FONCOB(IPC)
      IMPLICIT INTEGER (B-Z)
      COMMON /V1/ VILLE(100),X(5000),IX(5000),TAM(7)
      COMMON /V2/ AVILL(100,2),RPQ(100),IRPQ(100),TEMP(100)
      COMMON /V3/ IP(1000),IQ(1000),QANDU(1000),Y(1000)
      COMMON /V4/ NV,ASPQ,NPQ,NNIJ
      COMMON /V5/ IPT(100),ROUT(500),TROUT(500)
      COMMON /V6/ NBROUT,NAV,TYPAV(8)
      COMMON /V7/ NN
      COMMON /V10/ TVOL(500)
      DIMENSION ADC(200,6),ACTE(6),ABNBH(6)
      READ 107,(ACTE(I),ABNBH(I),I=1,NAV)
  107 FORMAT(2F8.1)
      DO 5 I=1,NAV
      DO 5 J=1,NBROUT
    5 ADC(J,I)=ACTE(I) + (ABNBH(I)/60.) * TVOL(J)
      WRITE(1,100)
  100 FORMAT(' LGL',T11,'FONCOB(F)')
      DO 1 IC=1,NPQ
      CY=Y(IC)
      P=IP(IC)
      Q=IQ(IC)
      CALL GENRPQ(P,Q)
      IF(NN.EQ.0) GO TO 6
      DO 2 JA=1,NN
    2 WRITE(1,101) P,Q,RPQ(JA),CY
  101 FORMAT(' AIJ',T11,'FONCOB(F)',PPOR,',',R3,R3,',',',R6,'=',I5)
    6 CALL GERCPQ(P,Q,IPC)
      IF(NN.EQ.0) GO TO 1
      DO 3 JA=1,NN
    3 WRITE(1,104) P,Q,RPQ(JA),CY
  104 FORMAT(' AIJ',T11,'FONCOB(F)',PPOC,',',R3,R3,',',',R6,'=',I5)
    1 CONTINUE
      DO 4 IC=1,NBROUT
      DO 4 JA=1,NAV
      ADC(IC,JA)= - ADC(IC,JA)
    4 WRITE(1,106) ROUT(IC),TYPAV(JA),ADC(IC,JA)
  106 FORMAT(' AIJ',T11,'FONCOB(F)',NRA,',',R6,',',',R4,'=',F10.2)
      RETURN
      END
```

```
      SUBROUTINE CONTR1(IPC)
      IMPLICIT INTEGER (B-Z)
      COMMON /V1/ VILLE(100),X(5000),IX(5000),TAM(7)
      COMMON /V2/ RVILL(100,2),RPQ(100),IRPQ(100),TEMP(100)
      COMMON /V3/ IP(1000),IQ(1000),OANDU(1000),Y(1000)
      COMMON /V4/ NV,ASPQ,NPQ,NNIJ
      COMMON /V6/ NBROUT,NAV,TYPAV(8)
      COMMON /V7/ NN
      DIMENSION IPC(100),SIEGE(6),RIJ(1000),ALFIJ(1000),RPETQ(1000)
      MD1=0
      DO 4 L=1,NPQ
      VP=IP(L)
      VQ=IQ(L)
    4 RPETQ(L)= SHIFT(VP,18).OR.VQ
      READ 100, (SIEGE(L),L=1,NAV)
  100 FORMAT(6I5)
      READ 101, (RIJ(L),ALFIJ(L),L=1,NNIJ)
  101 FORMAT(1X,R6,3X,F4.3)
      I=1
      DO 2 LL=1,NBROUT
      IDEB=I+1
      IFIN=I+IX(I)
      NIJ=IX(I) - 1
      DO 1 MM=1,NIJ
      IPV=I+MM
      IDV=IPV+1
      PV=X(IPV)
      DV=X(IDV)
      WRITE(1,104) PV,DV,X(I)
  104 FORMAT(' LGL=',I1,',CONTR1.',R3,R3,'.',R6,'(M)')
      WRITE(1,110) PV,DV,X(I),MD1
  110 FORMAT(' RHS=',I1,',CONTR1.',R3,R3,'.',R6,'(M),MD1=',I2)
      PVDV=SHIFT(PV,18).OR.DV
      CALL TROUVE(RIJ,NNIJ,PVDV,J)
      DO 3 L=1,NAV
      ACOEF=SIEGE(L) * ALFIJ(J)
    3 WRITE(1,105) PV,DV,X(I),X(I),TYPAV(L),ACOEF
  105 FORMAT(' AIJ=',I1,',CONTR1.',R3,R3,'.',R6,'(M),NBA.',R6,'.',R4,
     1'=',F6.2)
      ACOEF= -1.
      DO 1 M=IDEB,IPV
      JP=X(M)
      DO 1 N=IDV,IFIN
      JQ=X(N)
      CALL GENCPQ(JP,JQ,IPC)
      IF(NN.EQ.0) GO TO 5
      WRITE(1,108) PV,DV,X(I),JP,JQ,X(I),ACOEF
```

```
108 FORMAT(≠ AIJ≠,I1,≠CONTRI.≠,R3,R3,≠.≠,R6,≠(M),PTPQR.≠,R3,R3,≠.≠,
   1R6,≠=≠,F4.1)
  5 JPJQ=SHIFT(JP,18).OR.JQ
    CALL TROUVE(RPETQ,NPQ,JPJQ,J)
    IF(J.EQ.0) GO TO 1
    WRITE(1,109) PV,DV,X(I),JP,JQ,X(I),ACOEF
109 FORMAT(≠ AIJ≠,I1,≠CONTRI.≠,R3,R3,≠.≠,R6,≠(M),PPQR.≠,R3,R3,≠.≠,
   1R6,≠=≠,F4.1)
  1 CONTINUE
    I=IFIN+1
  2 CONTINUE
    RETURN
    END
```

```
      SUBROUTINE CONTR2(IPC)
      IMPLICIT INTEGER (B-Z)
      COMMON /V1/ VILLE(100),X(5000),IX(5000),TAM(7)
      COMMON /V2/ BVILL(100,2),RPQ(100),IRPQ(100),TEMP(100)
      COMMON /V3/ IP(1000),IQ(1000),QANDV(1000),Y(1000)
      COMMON /V4/ NV,ASPQ,NPQ,NNIJ
      COMMON /V5/ IPT(100),ROUT(500),IROUT(500)
      COMMON /V7/ NN
      ACF=1.
      ACFF= -1.
      MD1=0
      DO 1 LL=1,NPQ
      VP=IP(LL)
      VQ=IQ(LL)
      CALL GENCPQ(VP,VQ,IPC)
      IF(NN.EQ.0) GO TO 1
      WRITE(1,100) VP,VQ
  100 FORMAT(' LGL',I1,'CONTR2.',R3,R3)
      WRITE(1,107) VP,VQ,MD1
  107 FORMAT(' RHS',I1,'CONTR2.',R3,R3,',MD1= ',I2)
      DO 3 L=1,NN
      WRITE(1,106) VP,VQ,BVILL(L,1),BVILL(L,2),RPQ(L),ACF
  106 FORMAT(' AIJ',I1,'CONTR2.',R3,R3,',PPQC.',R3,R3,'.',R6,'=',F4.1)
    3 CONTINUE
      CALL GENRPQ(VP,VQ)
      DO 2 L=1,NN
      WRITE(1,104) VP,VQ,VP,VQ,RPQ(L),ACFF
  104 FORMAT(' AIJ',I1,'CONTR2.',R3,R3,',PIPQR.',R3,R3,'.',R6,'=',F4.1)
    2 CONTINUE
    1 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE CONTR3(IPC)
      IMPLICIT INTEGER (B-Z)
      COMMON /V1/ VILLE(100),X(5000),IX(5000),TAM(7)
      COMMON /V2/ AVILL(100,2),RPQ(100),IRPQ(100),TEMP(100)
      COMMON /V3/ IP(1000),IQ(1000),QANDD(1000),Y(1000)
      COMMON /V4/ NV,ASPQ,NPQ,NNIJ
      COMMON /V5/ IPT(100),ROUT(500),IROUT(500)
      COMMON /V6/ NBROUT,NAV,TYPAV(8)
      COMMON /V7/ NN
      COMMON /V8/ ATPQ(1000)
      COMMON /V9/ P1,Q1
      AACOEF= -1.
      MD1=0
      READ 100,ACF,APT
  100 FORMAT(2F4.2)
      DO 1 IAB=1,NPQ
      VP=IP(IAB)
      VQ=IQ(IAB)
      P1=VP
      Q1=VQ
      WRITE(1,101) VP,VQ
  101 FORMAT(' LGL',I1,'CONTR3.',R3,R3)
      WRITE(1,102) VP,VQ,MD1
  102 FORMAT(' RHS',I1,'CONTR3.',R3,R3,'.',MD1= ',I2)
      CALL GECRPQ(VP,VQ,6)
      A1=ASPQ
      CALL GCRCPQ(VP,VQ,IPC)
      A2=ASPQ
      ASPQ=A1+A2
      ABMIN=ASPQ - ASPQ*ACF
      ABMAX=ASPQ + ASPQ*ACF
      ATPQ(IAB)= (QANDD(IAB) / ASPQ) * APT
      WRITE(1,104) P1,Q1,P1,Q1,AACOEF
  104 FORMAT(' AIJ',I1,'CONTR3.',R3,R3,'.',SPQ,'.',R3,R3,'=',F4.1)
      WRITE(1,103) P1,Q1,P1,Q1,AACOEF
  103 FORMAT(' AIJ',I1,'CONTR3.',R3,R3,'.',SPPQ,'.',R3,R3,'=',F4.1)
      WRITE(1,105) P1,Q1,ABMIN,ABMAX
  105 FORMAT(' BND',I1,'INI1,SPQ,'.',R3,R3,'=',F8.3,',',',F8.3)
    1 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE CONTR4(IPC)
      IMPLICIT INTEGER (B-Z)
      COMMON /V1/ VILLE(100),X(5000),IX(5000),TAM(7)
      COMMON /V2/ BVILL(100,2),RPQ(100),IRPQ(100),TEMP(100)
      COMMON /V3/ IP(1000),IQ(1000),OANDD(1000),Y(1000)
      COMMON /V4/ NV,ASPQ,NPQ,NNIJ
      COMMON /V5/ IPT(100),ROUT(500),IROUT(500)
      COMMON /V6/ NBROUT,NAV,TYPAV(8)
      COMMON /V7/ NN
      COMMON /V8/ ATPQ(1000)
      AACOEF= -1.
      MD1=0
      DO 1 IC=1,NPQ
      VP=IP(IC)
      VG=IQ(IC)
      WRITE(1,100) VP,VG
  100 FORMAT(' LG',T11,'CONTR4.',R3,R3,'(M)')
      WRITE(1,103) VP,VG,MD1
  103 FORMAT(' RHS',T11,'CONTR4.',R3,R3,',MD1=',I2)
      WRITE(1,102) VP,VG,VP,VG,ATPQ(IC)
  102 FORMAT(' AIJ',T11,'CONTR4.',R3,R3,'(M).SPQ.',R3,R3,'=',F8.3)
      GO TO 6
      ATPPQ=ATPQ(IC) / 10.
      WRITE(1,101) VP,VG,VP,VG,ATPPQ
  101 FORMAT(' AIJ',T11,'CONTR4.',R3,R3,'(M).SPPQ.',R3,R3,'=',F8.3)
    6 CALL GENRPQ(VP,VG)
      IF(NN.EQ.0) GO TO 5
      DO 2 JA=1,NN
    2 WRITE(1,105) VP,VG,VP,VG,RPQ(JA),AACOEF
  105 FORMAT(' AIJ',T11,'CONTR4.',R3,R3,'(M).PPQR.',R3,R3,',',R6,'=',
     1F4.1)
    5 CALL GENRPQ(VP,VG,IPC)
      IF(NN.EQ.0) GO TO 1
      DO 3 JA=1,NN
    3 WRITE(1,107) VP,VG,VP,VG,RPQ(JA),AACOEF
  107 FORMAT(' AIJ',T11,'CONTR4.',R3,R3,'(M).PPQC.',R3,R3,',',R6,'=',
     1F4.1)
    1 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE CONTR5
      IMPLICIT INTEGER (B-Z)
      COMMON /V5/ IPT(100),ROUT(500),IROUT(500)
      COMMON /V6/ NBROUT,NAV,TYPAV(8)
      COMMON /V10/ TVOL(500)
      DIMENSION AU(6),WAA(6)
      READ 101, (AU(I),WAA(I),I=1,NAV)
  101 FORMAT(F6.1,I4)
      DO 1 JA=1,NAV
      WRITE(1,102) TYPAV(JA)
  102 FORMAT(' LGL',T11,'CONTR5.',R4,'(P)')
      AMD1=AU(JA)*WAA(JA)
      WRITE(1,104) TYPAV(JA),AMD1
  104 FORMAT(' RHS',T11,'CONTR5.',R4,'(P),MU1= ',F10.2)
      DO 1 IC=1,NBROUT
    1 WRITE(1,105) TYPAV(JA),ROUT(IC),TYPAV(JA),TVOL(IC)
  105 FORMAT(' AIJ',T11,'CONTR5.',R4,'(P),NRA.',R6,'.',R4,'==',IR
      RETURN
      END
```

```
      SUBROUTINE STRUCM(I,IPT,ROUT,IROUT,NOROUT,NBROUT)
      IMPLICIT INTEGER (B-Z)
      DIMENSION IPT(100),ROUT(500),IROUT(500)
      COMMON /V1/ VILLE(100),X(5000),IX(5000),TAM(7)
      COMMON /V2/ BVILL(100,2),RPQ(100),IRPQ(100),TEMP(100)
      COMMON /V3/ IP(1000),IQ(1000),DANDQ(1000),Y(1000)
      COMMON /V4/ NV,NSPQ,NPQ,NNIJ
      COMMON /V10/ TVOL(500)
      DO 1 J=1,NV
    1 IPT(J)=0
    7 II=I+1
      IC=I
      READ 102, IA,NVOL,TEMVOL,(TAM(N),N=1,IA)
  102 FORMAT(I3,2I5,7(1X,R3))
      IF(IA.EQ.0) GO TO 10
      NOROUT=NOROUT+1
      TVOL(NOROUT)=TEMVOL
      ENCODE( 6,1000,X(1)) NOROUT
 1000 FORMAT(2HRR,I4.4)
      X(I)=SHIFT(X(I),36)
      NBROUT=NBROUT+1
      ROUT(NBROUT)=X(I)
      IROUT(NBROUT)=NVOL
      IX(I)=IA
      IB=IA+I
      DO 2 N=1,IA
      IC=IC+1
    2 X(IC)=TAM(N)
      DO 3 J=II,IB
      DO 4 K=1,NV
      IF(VILLE(K).EQ.X(J)) GO TO 5
    4 CONTINUE
      PRINT 103,X(J)
  103 FORMAT(5X,20HERREUR POUR LA VILLE,4X,R3)
      STOP #ERREUR1#
    5 IX(J)=IPT(K)
      IPT(K)=I
    3 CONTINUE
      I=I+1+IA
      IF(I.LT.4990) GO TO 7
      PRINT 104
  104 FORMAT(///,25H    DEPASSEMENT D ESPACE )
      STOP #ERREUR2#
   10 PRINT 108,I
  108 FORMAT(///,20X,20HLA TAILLE DE X EST  ,I5)
      RETURN
      END
```

```
      SUBROUTINE GENRPQ(P,Q)
      IMPLICIT INTEGER (B-Z)
      COMMON /V1/ VILLE(100),X(5000),IX(5000),TAM(7)
      COMMON /V2/ BVILL(100,2),RPQ(100),IRPQ(100),TEMP(100)
      COMMON /V4/ NV,NSPQ,NRQ,NNIJ
      COMMON /V5/ IPT(100),ROUT(500),IROUT(500)
      COMMON /V6/ NBROUT,NAV,TYPAV(8)
      COMMON /V7/ NN
      NN=0
      CALL TROUVE(VILLE,NV,P,J)
      K=IPT(J)
   16 L=IX(K)
      J1=K+1
      J2=K+L
      DO 12 J=J1,J2
      IF(P.NE.X(J)) GO TO 12
      IF(J.EQ.J2) GO TO 13
      J3=J+1
      DO 14 M=J3,J2
      IF(Q.NE.X(M)) GO TO 14
      NN=NN+1
      RPQ(NN)=X(K)
      IRPQ(NN)=M-J3+1
      GO TO 13
   14 CONTINUE
   13 IF(IX(J).EQ.0) GO TO 15
      K=IX(J)
      GO TO 16
   12 CONTINUE
   15 RETURN
      END
```

```
      SUBROUTINE GERCPQ(P,Q,IPT)
      IMPLICIT INTEGER (B-Z)
      DIMENSION IPT(100)
      COMMON /V1/ VILLE(100),X(5000),IX(5000),TAM(7)
      COMMON /V2/ AVILL(100,2),RPQ(100),IRPQ(100),TEMP(100)
      COMMON /V4/ NV,ASPQ,NPQ,NNIJ
      COMMON /V7/ NN
      NN=0
      CALL TROUVE(VILLE,NV,P,J)
      K=IPT(J)
      IF(K.EQ.0) GO TO 10
   16 L=IX(K)
      J1=K+1
      J2=K+L
      IF(P.NE.X(J1).OR.Q.NE.X(J2)) GO TO 13
      NN=NN+1
      RPQ(NN)=X(K)
      IRPQ(NN)=J2-J1
   14 IF(IX(J1).EQ.0) GO TO 15
      K=IX(J1)
      GO TO 16
   13 J11=J1
      DO 12 J1=J11,J2
      IF(P.EQ.X(J1)) GO TO 14
   12 CONTINUE
   15 IF(NN.NE.0) GO TO 41
   10 PRINT 110,P,Q
  110 FORMAT(/,' AUCUNE ROUTE A CONNEXION ENTRE ',R3,' ET ',R3)
   41 RETURN
      END
```

```
      SUBROUTINE GENCPQ(P,Q,IPT)
      IMPLICIT INTEGER (B-Z)
      COMMON /V1/ VILLE(100),X(5000),IX(5000),TAM(7)
      COMMON /V2/ BVILL(100,2),RPQ(100),IRPQ(100),TEMP(100)
      COMMON /V4/ NV,NSPQ,NPQ,NNIJ
      COMMON /V7/ NN
      DIMENSION IPT(100)
      NN=0
      CALL TROUVE(VILLE,NV,P,J)
      K=IPT(J)
   16 L=IX(K)
      J1=K+1
      J2=K+L
      DO 12 J=J1,J2
      IF(P.NE.X(J)) GO TO 12
      IF(J.EQ.J2) GO TO 13
      J3=J+1
      IF(Q.NE.X(J3)) GO TO 13
      NN=NN+1
      RPQ(NN)=X(K)
      BVILL(NN,1)=X(J1)
      BVILL(NN,2)=X(J2)
   13 IF(IX(J).EQ.0) GO TO 15
      K=IX(J)
      GO TO 16
   12 CONTINUE
   15 IF(NN.NE.0) GO TO 11
      PRINT 108,P,Q
  108 FORMAT(/,' PAS DE CONNEXION DIRECTE POUR ',R3,' ET ',R3)
   11 RETURN
      END
```

```
      SUBROUTINE GECRPQ(P,Q,NARR)
      IMPLICIT INTEGER (B-Z)
      COMMON /V1/ VILLE(100),X(5000),IX(5000),TAM(7)
      COMMON /V2/ AVILL(100,2),RPQ(100),IRPQ(100),TEMP(100)
      COMMON /V4/ NV,ASPQ,NPQ,NNIJ
      COMMON /V5/ IPT(100),ROUT(500),IROUT(500)
      COMMON /V6/ NBROUT,NAV,TYPAV(8)
      COMMON /V7/ NN
      COMMON /V9/ P1,Q1
      DIMENSION IND(100),JND(100)
      NN=0
      CALL TROUVE(VILLE,NV,P,J)
      K=IPT(J)
   16 L=IX(K)
      J1=K+1
      J2=K+L
      DO 12 J=J1,J2
      IF(P.NE.X(J)) GO TO 12
      IF(J.EQ.J2) GO TO 13
      J3=J+1
      DO 14 M=J3,J2
      IF(Q.NE.X(M)) GO TO 14
      NN=NN+1
      RPQ(NN)=X(K)
      IND(NN)=K
      IRPQ(NN)=M-J3+1
      GO TO 13
   14 CONTINUE
   13 IF(IX(J).EQ.0) GO TO 15
      K=IX(J)
      GO TO 16
   12 CONTINUE
   15 ASPQ=0.
      DO 30 LL=1,NARR
      JJ=0
      DO 31 J=1,NN
      IF(IRPQ(J).EQ.LL) 32,31
   32 JJ=JJ+1
      TEMP(JJ)=RPQ(J)
      JND(JJ)=IND(J)
   31 CONTINUE
      IF(JJ.EQ.0) GO TO 30
      AA=AKM(LL)
      DO 33 IS=1,JJ
      IF(NARR.EQ.6) GO TO 40
      IF(IX(JND(IS)).EQ.2) GO TO 40
      GO TO 33
   40 RI=TEMP(IS)
      DO 34 KK=1,NAV
```

```
      WRITE(1,104) P1,Q1,RT,TYPAV(KK),AA
  104 FORMAT(' AIJ',I1,'CONTR3.',R3,R3.',NRA.',R6,'.',R4,'=',F4.2)
   34 CONTINUE
      CALL TROUVE(ROUT,NBROUT,RT,J)
      ASPQ=ASPQ+IROUT(J)*AA
   33 CONTINUE
   30 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE GCRCPQ(P,Q,IPT)
      IMPLICIT INTEGER (B-Z)
      DIMENSION IPT(100)
      COMMON /V1/ VILLE(100),X(5000),IX(5000),TAM(7)
      COMMON /V2/ BVILL(100,2),RPQ(100),IRPQ(100),TEMP(100)
      COMMON /V4/ NV,ASPQ,NPQ,NNIJ
      COMMON /V7/ NN
      COMMON /V9/ P1,Q1
      DIMENSION IND(100),JND(100),KRPQ(100)
      NN=0
      CALL TROUVE(VILLE,NV,P,J)
      K=IPT(J)
      IF(K.EQ.0) GO TO 10
   16 L=IX(K)
      J1=K+1
      J2=K+L
      IF(P.NE.X(J1).OR.Q.NE.X(J2)) GO TO 13
      NN=NN+1
      RPQ(NN)=X(K)
      IND(NN)=K
      IRPQ(NN)=J2-J1
   14 IF(IX(J1).EQ.0) GO TO 15
      K=IX(J1)
      GO TO 16
   13 J11=J1
      DO 12 J1=J11,J2
      IF(P.EQ.X(J1)) GO TO 14
   12 CONTINUE
   15 IF(NN.NE.0) GO TO 41
   10 PRINT 110,P,Q
  110 FORMAT(/,' AUCUNE ROUTE A CONNEXION ENTRE ',R3,' ET ',R3)
      ASPQ=0.
      GO TO 11
   41 AV=0.
      DO 50 LL=1,NN
   50 KRPQ(LL)=IRPQ(LL)
      DO 30 LL=2,3
      JJ=0
      DO 31 J=1,NN
      IF(KRPQ(J).NE.LL) GO TO 31
      JJ=JJ+1
      TEMP(JJ)=RPQ(J)
      JND(JJ)=IND(J)
   31 CONTINUE
      IF(JJ.EQ.0) GO TO 30
```

```
      IF(LL.NE.2) GO TO 32
      DO 33 IS=1,JJ
      P=X(JND(IS)+1)
      Q=X(JND(IS)+2)
      CALL GECRPQ(P,Q,1)
      ACUM1=ASPQ
      P=Q
      Q=X(JND(IS)+3)
      CALL GECRPQ(P,Q,1)
      ACUM2=ASPQ
      AKC=AMIN1(ACUM1,ACUM2)
   33 AV=AV+AKC
      GO TO 30
   32 DO 34 IS=1,JJ
      P=X(JND(IS)+1)
      Q=X(JND(IS)+2)
      CALL GECRPQ(P,Q,1)
      ACUM1=ASPQ
      P=Q
      Q=X(JND(IS)+3)
      CALL GECRPQ(P,Q,1)
      ACUM2=ASPQ
      P=Q
      Q=X(JND(IS)+4)
      CALL GECRPQ(P,Q,1)
      ACUM3=ASPQ
      AKC=AMIN1(ACUM1,ACUM2,ACUM3)
   34 AV=AV+AKC
   30 CONTINUE
      ASPQ=AV
   11 RETURN
      END
```

```
      SUBROUTINE TROUVE(C,NV,P,J)
      IMPLICIT INTEGER (B-Z)
      DIMENSION C(NV)
      K1=1
      K2=NV
    5 IF(K1.LE.K2) GO TO 6
      J=0
      RETURN
    6 L=(K1+K2)/2
      IF(P-C(L)) 2,3,4
    3 J=L
      RETURN
    2 K2=L-1
      GO TO 5
    4 K1=L+1
      GO TO 5
      END
```

```
      FUNCTION AKM(LL)
      GO TO (1,2,3,4,5,6),LL
    1 AKM=1.
      RETURN
    2 AKM=.9
      RETURN
    3 AKM=.85
      RETURN
    4 AKM=.8
      RETURN
    5 AKM=.75
      RETURN
    6 AKM=.7
      RETURN
      END
```